Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Bojan Robba

Pregled odprtokodnih orodij za testiranje spletnih aplikacij v Python-u

diplomsko delo

visokošolski strokovni študijski program prve stopnje računalništvo in informatika

Mentor: viš. pred. dr. Igor Rožanc

Ljubljana 2015

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Pregled odprtokodnih orodij za testiranje spletnih aplikacij v Python-u

Tematika naloge:

V diplomski nalogi predstavite in ovrednotite več odprtokodnih orodij za testiranje spletnih aplikacij, ki so napisane v programskem jeziku Python, in sicer tako orodja za testiranje izdelanih spletnih aplikacij kot orodja za preverjanje pravilnosti napisane kode. V ta namen najprej določite principe in kriterije za izvedbo primerjave, nato pa primeren izbor orodij opišite in na podlagi izkušenj ocenite. Končen rezultat naloge naj bo izbor testnih orodij, ki omogoča razvijalcu učinkovito testiranje tovrstnih aplikacij.

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Bojan Robba sem avtor diplomskega dela z naslovom:

Pregled odprtokodnih orodij za testiranje spletnih aplikacij v Python-u

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom viš. pred. dr. Igorja Rožanca,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela na svetovnem spletu preko univerzitetnega spletnega arhiva.

V Ljubljani, dne 28. avgusta 2015 Podpis avtorja:

Zahvaljujem se viš. pred. dr. Igorju Rožancu za vso pomoč ter družini za potrpljenje in pomoč skozi leta šolanja in pisanja diplome.

Kazalo

Slike

Seznam uporabljenih kratic

kratica	angleško	slovensko
URL	Uniform Resource Locator	enolični krajevnik vira
HTTP	HyperText Transfer Protocol	protokol za prenos hiperteksta
HTTPS	HTTP Secure	protokol za varni prenos hiperteksta
HTML	Hyper Text Markup Language	jezik za označevanje nadbesedila
SSL	Secure Sockets Layer	sloj varnih vtičnic
TLS	Transport Layer Security	sloj varnega prenosa
PDF	Portable Document Format	prenosna oblika dokumenta

Povzetek

Namen diplomske naloge je pregledati in oceniti različna orodja za pregledovanje in testiranje spletnih aplikacij v programskem jeziku Python.

Orodja Twill, Webunit in FunkLoad so namenjena testiranju končnih spletnih aplikacij. Glavna prednost orodja Twill je enostavna uporaba, slabost pa zastarelost in majhna razširjenost. Orodje Webunit je tudi enostavno, vendar ni prilagojeno za uporabo v Windows okolju. Orodje FunkLoad je nadgradnja orodja Webunit s snemalnikom za pripravo testov.

Orodja PyChecker, Pylint in Pyflakes so namenjena za pregledovanje pravilnosti kode. PyChecker najde vse napake v kodi, vendar pri tem izvede kodo. Pyflakes je nadgraditev orodja PyChecker, ki kode ne izvede, vendar ne odkrije vseh napak. Pylint poleg napak odkriva tudi lepopisne napake v kodi in je ne izvede.

Na koncu je predstavljen nabor orodij iz različnih področij, ki bi jih bilo najbolje uporabljati pri svojih projektih.

Ključne besede: Python, spletne aplikacije, testiranje, preverjanje kode.

Abstract

The objective of my thesis is to review and evaluate different tools for inspecting and testing Python web applications.

Tools Twill, Webunit and FunkLoad are intended for testing web applications. Twill is easy to use, but it is outdated and unpopular. Tool Webunit is also easy to use, but it is not adapted for usage in Windows. Tool FunkLoad is upgraded from Webunit with a recorder for making tests.

Tools PyChecker, Pylint in Pyflakes are intended for checking source code. PyChecker detects all errors in code, but needs to execute the code. Pyflakes is upgraded from PyChecker and does not need to execute the code, but does not detect all errors. Pylint, beside errors, also detects poorly formed lines inside code.

At the end is presented a set of tools from different fields, which are the most suitable for usage in our projects.

Keywords: Python, web applications, testing, code checking.

Poglavje 1

Uvod

Testiranje programske opreme je poznano že desetletja, vendar se je v tem času pristop zelo spremenil. V preteklosti so to izvajali pripravniki, študentje ali na novo zaposleni uslužbenci, ki so sedeli pred računalnikom in na različne načine izvajali program ali aplikacijo. Tak način odkrivanja napak je bil časovno in denarno zelo potraten in ne vedno zanesljiv. Razvoj testnih orodij (ang. testing tools) je postopek testiranja pohitril, pocenil in mu povečal zanesljivost.

Testna orodja so izbirke programskih izdelkov, zasnovanih posebej za pomoč izvajalcem in upravljalcem pri programskih testiranjih z različnih vidikov projektnega testiranja [1]. Večina testnih primerov deluje po principu izvedbe nekega dejanja in preverjanja, ali je dani rezultat enak pričakovanemu. Test je uspešen, če smo dobili pričakovan rezultat, v nasprotnem primeru pa neuspešen.

Cilj diplomskega dela je poiskati in pregledati več obstoječih testnih orodji na spletu, izbrati nekaj najboljših in med njimi poiskati najprimernejšega na določenem področju, katerega bomo lahko uporabljali v svojih projektih. Omejili se bomo na spletna testna orodja, ki so napisana v programskem jeziku Python, predvsem na orodja za spletno testiranje (ang. Web Testing Tools) ter na orodja za preverjanje napak v programski kodi (ang. Source Code Checking Tools). Iz vsakega od obeh področji bomo pregledali orodja,

ki so nam na voljo na spletu in med njimi izbrali nekaj orodij, jih med seboj primerjali in izdelali svoj nabor orodij.

1.1 Delovno okolje

Da bi bilo testiranje objektivno, bomo pri testiranju testnih orodij uporabljali enako testno okolje:

- Microsoft Windows 7 Professional 64-bit in Ukazna vrstica ali konzola (ang. CMD); Čeprav večina razvijalcev spletnih aplikacij in testnih orodij pri svojem delu uporablja Linux ali iOS (ki izvirata iz Unix jedra), nas zanima usklajenost testnih orodij z Windows okoljem.

- Python 2.7.10 [2]; Kljub novejši verziji je še vedno najpogosteje uporabljen. Zaradi velikih sprememb v sintaksi jezika, nekatera orodja še ne delujejo v Pythonu 3.x.x.

- Dodatno bomo uporablili še Beležnico (ang. Notepad) in urejevalnik kode IDLE, ki se namesti skupaj s Python jezikom.

1.2 Pregled

V naslednjem poglavju bomo omenili tudi orodje Unittest [3], ki je sicer osnova za testiranje ne glede na področje programiranja; za spletne aplikacije, igre, mobilne aplikacije itd. Nekatera testna orodja ga uporabljajo pri preverjanju enakosti pričakovanih vrednosti.

Nato si bomo ogledali testna orodja za testiranje končnih spletnih aplikacij ter jih na preprostem primeru preverili in podali svojo oceno ter mnenje. Ocenjevanje bo potekalo s pomočjo Likertove lestvice (ang. Likert scale).[4] Orodja bomo ocenjevali na podlagi njihove spletne predstavitve oziroma dokumentacije ter dejanske uporabe. Pri uporabi bo večjo težo ocene prispevalo

pisanje testov, uporaba testov in poročila o uspešnosti ali neuspešnosti testa, manjšo težo pa sama namestitev in priprava orodja za uporabo. Vsa predstavljena orodja so brezplačna in dostopna na spletu.

V tretjem poglavju bomo predstavili orodja za pregledovanje programske kode. Ocenili bomo namestitev in pripravo orodja za delovanje ter jih na dveh primerih preizkusili. Ocenili jih bomo tudi glede na število najdenih napak in na razvidnost napak iz poročila.

V zaključku vsakega poglavja bomo pregledali ocene orodij ter na koncu diplomskega dela predstavili lasten nabor izbranih orodij, ki so najprimernejša za uporabo.

Poglavje 2

Orodja za testiranje spletnih aplikacij

Spletna testna orodja so namenjena testiranju končne aplikacije s simulacijo spletnega brskalnika oziroma uporabnika. Za razliko od orodij, ki se osredotočajo na testiranje enot (ang. unit testing), le-ta običajno dinamično testirajo končno aplikacijo. Glede na delovanje jih delimo na dve kategoriji [5]:

- orodja za simulacijo spletnega brskalnika (ang. Browser simulation tools)
- orodja za avtomatizacijo spletnega brskalnika (ang. Browser automation tools)

Orodja za simulacijo spletnega brskalnika uporabljajo protokola HTTP in HTTPS za pošiljanje zahtev strežniku in analizo njegovih odgovorov. Omogočajo izpolnjevanje tekstovnih polj (ang. text field) in obrazcev (ang. form), klikanje na povezave, gumbe in potrditvena polja (ang. check box), sledenjem preusmeritvam (ang. redirect), preverjanje kode HTTP odgovorov ter zajemanje HTML kode. Na tem področju si bomo ogledali večje število orodij in med njimi izbrali le ožji izbor orodij za predstavitev.

Orodja za avtomatizacijo spletnega brskalnika pa izvajajo testiranje spletne strani preko brskalnika. Za pripravo testov moramo orodje le zagnati znotraj spletnega brskalnika, orodje nato sledi dogajanju in beleži vsak dogodek, ki ga izvajamo. Tako posnet test se zažene znotraj brskalnika in med izvajnjem lahko v brskalniku opazujemo dogajanje. Tudi na tem področju bomo pregledali večje število orodij, predstavili pa ožji izbor.

2.1 Postopek testiranja in ocenjevanja

Domača stran, dokumentacija in navodila

Prvi del testiranja bo obsegal pregled spletne strani in dokumentacije orodja. Pregledali bomo, kako je predstavljeno samo orodje ter njegove metode. Ocenjevali bomo po naslednjih glavnih kriterijih:

- Domača stran in dokumentacija morata biti pregledno napisani, da lahko brez težav najdemo kar potrebujemo.
- Seznam metod mora obsegati vse metode, ki jih določeno orodje ponuja. Te metode morajo biti opisane in predstavljene s praktičnimi primeri.
- Naveden mora biti seznam najpogostejših napak z rešitvami. Tu je zaželjena še poveza do diskusij (forum, strani namenjene poročanju napak, elektronska sporočila itd.).

Namestitev

Tu si bomo ogledali sam postopek namestitve ter priprave orodja za pisanje testov in testiranje. Ocenjevali bomo po naslednjih glavnih kriterijih:

- Postopek namestitve mora biti opisan na spletni strani. Če za uporabo orodja potrebujemo namestiti še dodatne pakete, morajo biti navedeni skupaj z navodili za namestitev.

- Orodje mora po namestitvi biti pripravljeno za delovanje. V nasprotnem primeru mora orodje nuditi enostavno prilagoditev. Postopek in razlogi za prilagoditev morajo biti navedeni na spletni strani.

Izdelava testnih primerov

Tu bomo izdelali testni primer za določeno spletno aplikacijo. Za pisanje bomo uporabili dogovorjeni program, razen če orodje ponuja lasten urejevalnik kode ali drugi način priprave testov. Ocenjevali bomo po naslednjih glavnih kriterijih:

- Iz imena metode orodja mora biti jasno razbrano kaj metoda dela.
- Metode morajo biti enotno in nedvoumno poimenovane.
- Orodje mora nuditi dodatne načine priprave testnih primerov, ki se razlikujejo od tradicionalnega pisanja.

Testiranje

Tu bomo izdelani testni primer izvedli in si ogledali delovanje orodja. Izvedli bomo dva testa, prvi bo uspešen, pri drugemu pa bomo spremenili ime končne strani in si ogledali izvajanje v primeru neuspešnosti testa. Ocenjevali bomo po naslednjih glavnih kriterijih:

- Orodje mora nuditi dodatne nastavitve za testiranje, na primer hitrost izvajanja zaporednih testov.
- Delovati mora hitro in brez napak, ki niso posledica testnih primerov ali aplikacije.
- Orodje mora med delovanjem prikazovati povratne informacije uporabniku preko konzole. Najpomembnejši informaciji sta vrstica in metoda, ki jo orodje trenutno izvaja.

Poročila

Po končanem testiranju si bomo ogledali še izpisana poročila uspešnih in neuspešnih testov. Ocenjevali bomo po naslednjih glavnih kriterijih:

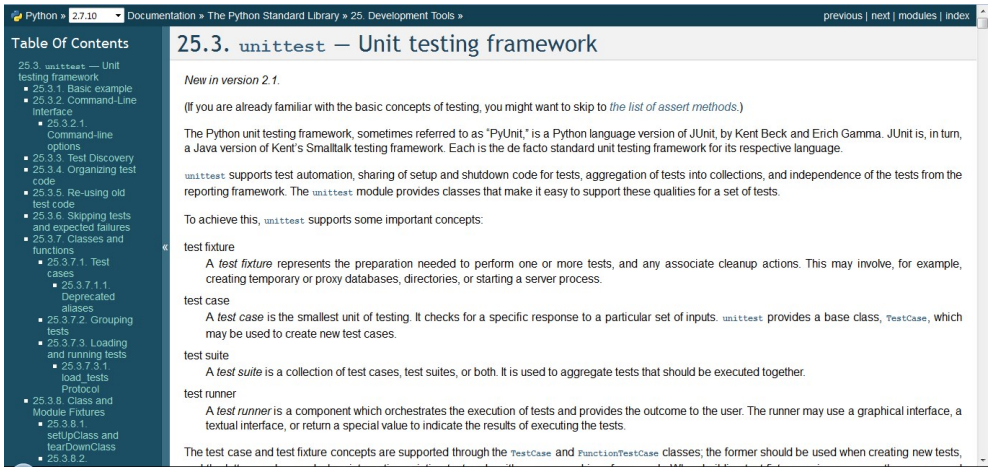
- Jasen izpis uspešnosti testa. V primeru neuspeha mora izpisati vrstico napake in javiti dejansko vrednost.
- Poročila (ne glede na uspeh) mora izpisati tudi v datoteko na določeno mesto.

2.2 Orodje Unittest

Unittest (imenovan tudi PyUnit) je osnovno testno orodje in je prvo orodje, ki je bilo vgrajeno v Python standardno knjižnico (Python verzija 2.1.). Na sliki (2.1) je prikazana spletna dokumentacija.[3] Namenjen je testiranju enot, kar pomeni, da lahko testira dele kode neodvisno od končnega izdelka in neodvisno od uporabniškega vmesnika. Tako lahko testiramo določeno metodo programa, ne da bi imeli celoten program (imeti moramo le dele, katere potrebuje dana metoda).

Testno orodje Unittest je bilo izdelano po principu JUnit testnega orodja [6], zato se v osnovi ne razlikuje veliko od ostali xUnit orodji (JUnit v Javi, CUnit v C-ju [7], NUnit v C# [8], itd.) Prav zato se kot glavno testno orodje Pythona pogosto uporablja skupaj z drugimi testnimi orodji (na primer: FunkLoad ga uporablja pri svojih testih za preverjanje enakosti).

Da lahko napišemo določeno funkcijo v programu, moramo poznati argumente, ki jih funkcija prejme, kaj mora z njimi narediti ter argumente, ki jih vrne kot odgovor. To je tudi osnova za pisanje testov za določeno funkcijo. Unittest namreč omogoča uporabniku, da pripravi teste po principu: če pošljem v funkcijo določene argumente, pričakujem določen odgovor. Naloga Unittesta je, da to preveri in potrdi (test je uspešen) ali ovrže (test je neuspešen).



Slika 2.1: Spletna stran orodja Unittest

2.3 Programski jezik Twill

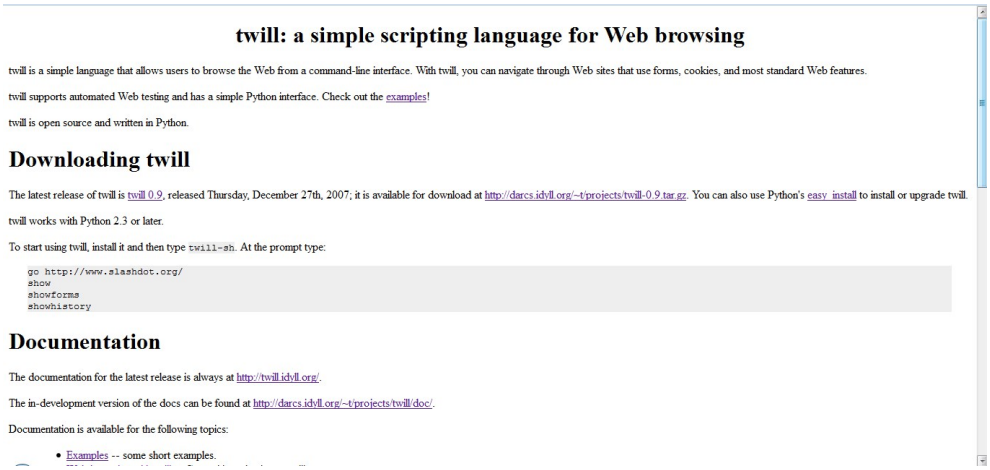
Twill je preprost programski jezik, ki omogoča brskanje po spletnih straneh preko ukazne vrstice.[9] Vključuje piškotke (ang. cookies), osnovne avtorizacije in skoraj celoten HTTP protokol, vendar ne podpira JavaScript-a [10]. Twill nam tako omogoča obiskovanje in navigacijo po spletni strani, iskanje in sledenje spletnim povezavam in gumbom, izpolnjevanje obrazcev ter uporabo spletnih piškotkov.

Glavni avtor je C. Titus Brown, ki ga je na spletu objavil leta 2005, nato pa ga je nadgrajeval s pomočjo uporabnikov, ki so ga obveščali o napakah ali ga dopolnjevali z lastno kodo. Zadnja uradna verzija programskega jezika je 0.9, ki je bila izdana 27. decembra 2007. Jezik je na voljo za uporabo, predelavo in distribucijo pod licenco MIT.

Twill lahko uporabljamo tudi znotraj Pythonovih programov, enako kot bi uporabljali standardne knjižnice (ang. library). Z ukazom:

```
from twill.commands import *
```

lahko uvozimo vse ukaze in jih uporabljamo znotraj Pythonove kode. Na tak način lahko testne primere še izboljšamo, saj jim lahko dodamo Pythonove ukaze in tako lahk zanaliziramo celotno HTML kodo.



Slika 2.2: Spletna stran programskega jezika Twill

2.3.1 Pregled in ocenjevanje

Domača stran, dokumentacija in navodila

Spletna stran testnega jezika Twill (Slika 2.2) je narejena minimalistično, vendar vključuje vse osnove, ki jih potrebujemo za razumevanje in uporabo Twilla. Na strani najdemo seznam vseh metod, ki jih orodje ponuja. Vsaka metoda je predstavljena z imenom, seznamom argumentov ter opisom delovanja. Naveden je tudi seznam vseh razširitvenih modulov, ki so vključena v paketu Twill, seznam znanih napak in hroščev (ang. bugs) ter arhiv elektronskih pisem, v katerih so uporabniki razvijalcu pošiljali napake in vprašanja. Z nekaj osnovnimi primeri je predstavljen tudi postopek uporabe Twilla v Pythonu. Čeprav so vsa navodila na strani zelo na kratko napisana, so dovolj razumljiva. Na spletni strani tudi ni omenjena knjiga o jeziku Twill [11], katere soavtor je C. Titus Brown.

Namestitev

Testni jezik Twill lahko namestimo preko ukazne vrstice (Windows: CMD, Linux: Terminal) z ukazom `easy_install`. Namestitev je hitra. Twill se avtomatsko namesti tudi k seznamu Pythonovih skript (v mapo `...\Python27`

\Scripts), zato nam ni potrebno dodajati nove vrednosti v sistemsko spremenljivko Path; zadostuje že obstoječa pot do Pythonovih skript.

Izdelava testnih primerov

Testne primere lahko pišemo v vsakem urejevalniku, ki omogoča pisanje programske kode. Obseg Twill metod je majhen, zato lahko z njim testiramo samo osnovne strani. Vendar so le te enostavne za uporabo, predvsem za nekoga, ki se prvič srečuje s testiranjem spletnih strani. Napisane primere preprosto shranimo kot Pythonovo kodo (.py).

Koda 2.1: Twill testni primer

```
go http://www.mojtestniforum.net/ #ce nisi prijavljen te preusmeri
    na stran za prijavo
title 'Log in' #preveri ali je prislo do pravilne preusmeritve
fv 1 username uporabnik
fv 1 password gesslo
submit login
title 'Moj Testni Forum' #preveri ali smo na pravilni strani
```

Pisanje testnih primerov je enostavno, saj lahko v ukazni vrstici Twill poženemo kot brskalnik in te ukaze ročno vpisujemo v vrstico. Tako lahko ročno sledimo vsakemu ukazu. Kljub temu je pisanje dolgih testnih primerov zelo počasno.

Testiranje in poročila

Testni primer zaženemo v ukazni vrstici z ukazom `twill-sh` in kot argument navedemo testni primer. Twill nas bo na vsakem koraku obveščal v kateri vrstici testnega primera se nahaja in pri določenih metodah tudi izpisoval dodatne informacije. V primeru napake, to je, ko se dejanska vrednost razlikuje od pričakovane, se bo testiranje v trenutku končalo.

Spodaj sta navedeni dve poročili. Uspešno poročilo je izvedba testa 2.1 omenjenega pri izdelavi testnih primerov, pri drugem poročilu pa smo pre-

verjali naslov (ukaz `title`) prve strani foruma in ne imena strani s prijavo. Orodje ne ponuja ukaza za pričakovano napako, kot je pri Unittestu ukaz `@unittest.expectedFailure`.

Koda 2.2: Poročilo, ko se testni primer izvede pravilno

```
>> EXECUTING FILE forumOK.py
AT LINE: forumOK.py:0
==> at http://www.mojtestniforum.net/login
AT LINE: forumOK.py:1
title is 'Log in'.
AT LINE: forumOK.py:2
AT LINE: forumOK.py:3
AT LINE: forumOK.py:4
Note: submit is using submit button: name="login", value="Log in"

AT LINE: forumOK.py:5
title is 'Moj Testni Forum'.
--
1 of 1 files SUCCEEDED.
```

Koda 2.3: Poročilo, ko se tesni primer izvede z napako

```
>> EXECUTING FILE forumFail.py
AT LINE: forumFail.py:0
==> at http://www.mojtestniforum.net/login
AT LINE: forumFail.py:1
title is 'Log in'.
** UNHANDLED EXCEPTION: title does not contain 'Moj Testni Forum'
--
0 of 1 files SUCCEEDED.
Failed:
   forumFail.py
```

Poročilo se izpiše samo v konzoli in ne ustvarja dodatnih datotek s poročilom,

kar povzroči pri velikem številu testov težave. Kljub temu je poročilo dovolj jasno, da se iz njega lahko razbere postopek testa, v primeru napake pa tudi v kateri vrstici je prišlo do nje in kakšna je bila dejansko vrednost.

Končna ocena

- Domača stran, dokumentacija in navodila - Ocena: 3

  Pozitivno: Ukazi so predstavljeni z opisom, navedena so sporočila spletne pošte s poročili napak ter odgovori.

  Negativno: Nekoliko nepregledno, nima kazala, ima le nekaj podstrani, malo število praktičnih primerov.

- Namestitev - Ocena: 5

  Pozitivno: Namestitev je hitra in enostavna, postopek namestitve je naveden na strani.

- Izdelava testnih primerov - Ocena: 3

  Pozitivno: Kratki ukazi, enostavni za razumevanje.

  Negativno: Pisanje dolgih testov je dolgotrajno, saj je potrebno izdelovati teste ročno.

- Testiranje - Ocena: 4

  Pozitivno: Testiranje poteka hitro, v konzoli se izpisujejo vrstice, katere trenutno izvaja ter dodatne informacije.

- Poročila - Ocena: 3

  Pozitivno: Ob napakah navede pričakovano vrednost ter dobljeno vrednost, navede seznam neuspelih testov.

  Negativno: Poročilo se izpiše samo v konzolo.

Testno orodje je enostavno za uporabo, predvsem za nekoga, ki se prvič sooča s testiranjem, vendar ni primerno za večje projekte. Druga razloga

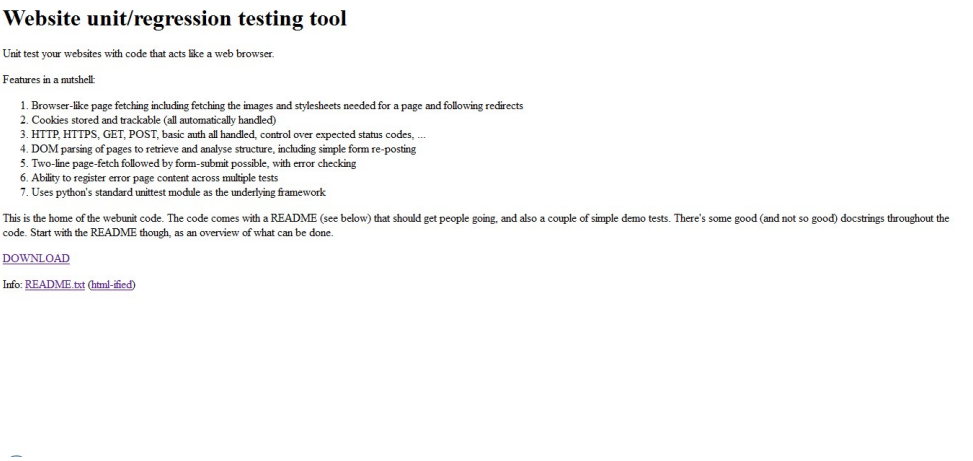
sta tudi zastarelost, saj orodje ni bilo posodobljeno že dobrih 8 let. Tudi poročanje napak še vedno poteka preko elektronske pošte in ne preko spletnih strani.

2.4 Orodje Webunit

Webunit je testno orodje, ki je napisano v programskem jeziku Python.[12]. Teste enako kot prej pišemo s pomočjo Python jezika in Unittest orodja. Deluje lahko na dva načina:

- Način lovljenja (ang. fetch): Privzeti način zajema le HTML kodo.
- Način strani (ang. page): Ta način deluje podobno kot spletni brskalniki, saj pri odpiranju strani naloži tudi slike in oblikovne predloge (ang. stylesheet).

Webunit je odprtokodno orodje. Trenutna verzija je 1.3.10, ki je bila izdana 5. maja 2010. To je tudi zadnja vidna aktivnost na spletni strani. Na osnovi Webunit orodja se je razvilo novo testno orodje imenovano FunkLoad (več o tem orodju v poglavju 2.5).


Website unit/regression testing tool

Unit test your websites with code that acts like a web browser.

Features in a nutshell:

1. Browser-like page fetching including fetching the images and stylesheets needed for a page and following redirects
2. Cookies stored and trackable (all automatically handled)
3. HTTP, HTTPS, GET, POST, basic auth all handled, control over expected status codes, ...
4. DOM parsing of pages to retrieve and analyse structure, including simple form re-posting
5. Two-line page-fetch followed by form-submit possible, with error checking
6. Ability to register error page content across multiple tests
7. Uses python's standard unittest module as the underlying framework

This is the home of the webunit code. The code comes with a README (see below) that should get people going, and also a couple of simple demo tests. There's some good (and not so good) docstrings throughout the code. Start with the README though, as an overview of what can be done.

DOWNLOAD

Info: README.txt (html-ified)

Slika 2.3: Spletna stran orodja Webunit

2.4.1 Pregled in ocenjevanje

Domača stran, dokumentacija in navodila

Domača stran (Slika 2.3) je narejena minimalistično. Ob namestitvi Webunit orodja dobimo vsebino spletne strani prepisano v `readme.txt` datoteko. Na strani ne najdemo nobenega načina za poročanje napak ali pomoči uporabnikom. Navedeni sta le povezavi na dve spletni strani, komur pripadajo avtorske pravice.

Namestitev

Postopek za namestitev na spletni strani ni napisan. Stran vsebuje samo povezavo do strani s prenosom. Vendar, ker gre za testno orodje napisano v Python jeziku, ga lahko namestimo preko ukazne vrstice z ukazom `easy_install`.

Problem

Ukaz za zagon testov je `./run_tests`, ki pa v okolju Windows ne deluje. Poskušali smo poiskati na spletu in v datotekah testnega orodja način, kako bi lahko drugače zagnali teste, vendar nismo našli ustrezne rešitve. To nam je tudi onemogočilo nadaljnjo testiranje orodja. Vseeno pa sem ga vključil v svoje diplomsko delo, ker je osnova orodja FunkLoad.[13]

Končna ocena

- Domača stran, dokumentacija in navodila - Ocena: 2

  Pozitivno: Ukazi so predstavljeni z opisom, navedenih je nekaj primerov.

  Negativno: Nepregledno, nima kazala, ima le dve strani (uvodno stran in dokumentacijo), malo število praktičnih primerov.

- Namestitev - Ocena: 2

Pozitivno: Namestitev je hitra in enostavna, postopek namestitve je naveden na strani.

Negativno: Ob namestitvi orodje ni primerno za Windows okolje, na spletni strani ni navedene prilagoditve.

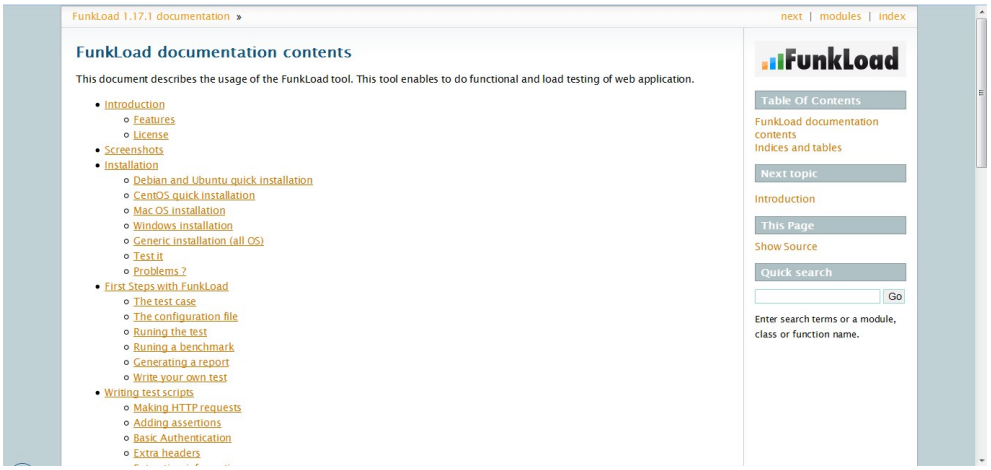
2.5 Orodje FunkLoad

FunkLoad je testno orodje, ki je napisano v programskem jeziku Python ter testne primere pišemo s pomočjo orodja Unittest.[13] Za preverjanje enakosti uporablja Unittest ukaz `assert_`. Razvilo se je iz orodja Webunit (več v poglavju 2.4), kateremu so dodali dodatne funkcije in izboljšave.

Uporabljamo ga lahko za:

- funkcionalno in regresijsko testiranje spletnih projektov,
- testiranje zmogljivosti spletnega strežnika, na katerem deluje spletna aplikacija, [14]
- testiranje spletne aplikacije pod veliko obremenitvijo (veliko število uporabnikov, veliko klicev v bazo podatkov...) in testiranje obnovljivosti aplikacije,
- pisanje spletnih agentov.

FunkLoad omogoča uporabo HTTP in HTTPS protokola, SSL in TLS s ključem in certifikatom, HTTP posredniškega strežnika (ang. proxy server), spletnih piškotkov ter posnema spletni predpomnilnik (ang. browser cache). Omogoča tudi pridobivanje CSS, javascript in slik s spletnih strani, za katere tudi meri časovno porabo prenosa. Ponuja tudi napredne funkcije, kot so prikazovanje spletnih strani v realnem času na brskalniku, preverjanje delovanja ene ali več strani med samim izvajanjem testov, vključevanje in izključevanje testov s pomočjo regex notacije, podpira pyUnit teste in doctest, itd.



Slika 2.4: Spletna stran orodja FunkLoad

FunkLoad je brezplačno programsko testno orodje, izdano pod licenco GNU GPL. Prva uradna verzija (1.0.0) je bila v javnost izdana 1. septembra 2005 s strani podjetja Nuxeo. Zadnja verzija (1.17.1) je bila izdana 6. maja 2015, kar nakazuje, da testno orodje redno razvijajo in nadgrajujejo. Nadgrajevanje je potekalo s pomočjo uporabnikov, ki so pošiljali poročila o napakah ali idejah. Prav tako so od uporabnikov in prostovolnih razvijalcev prejemali razširitve z novimi ali nadgradnjami starih metod orodja.

2.5.1 Pregled in ocenjevanje

Domača stran, dokumentacija in navodila

Uradna stran (Slika 2.4) služi kot uradna dokumentacija. Tu lahko najdem navodila za namestitev na petih različnih operacijskih sistemih (Debian, Ubuntu, CentOs, Mac OS in Windows) ter seznam možnih napak, na katere lahko med namestitvijo naletimo in rešitev, kako jih odpravimo. Ponuja nam tudi osnovna in s primeri dopolnjena navodila, ki služijo kot uvod v orodje FunkLoad ter predstavitev naprednejših metod. Vse metode so predstavljene na praktičnih primerih z dodanimi komentarji in opisi. Na strani je tudi naveden seznam najpogostejših vprašanj ter povezav do blogov, kjer je FunkLoad

predstavljen ali uporabljen skupaj z drugimi orodji. Celotna vsebina strani je na voljo tudi v PDF formatu na uradni strani.

Namestitev

Namestitev orodja FunkLoad je predstavljena na spletni strani skupaj z vsemi dodatnimi paketi, ki jih moramo namestiti za pravilno delovanje orodja FunkLoad. Celotna namestitev poteka preko `easy_install` ukaza. Tudi to orodje namesti svoje ukaze v mapo`...\Python27\Scripts`) in je ob namestitvi že pripravljeno za uporabo.

Izdelava testnih primerov

Testne primere lahko izdelujemo na dva možna načina. Prvi je standardni način, odpremo program za pisanje kode in vanj zapisujemo vrstico po vrstico, kaj naj testno orodje dela na spletni strani. Drugi način je, da posnamemo testni primer. Za snemanje potrebujemo namestiti TCPWatch Python proxy. Navodila za namestitev TCPWatch proxy strežnika so navedena na spletni strani FunkLoada. Le to orodje omogoča, da v spletnem brskalniku nastavimo proxy strežnik na localhost in vrata (ang. port) na 8090. Nato v ukazni vrstici samo zaženemo ukaz: `fl-record prviTest`

Spletni brskalnik bo preko navedenega strežnika dostopal do strani, orodje pa bo sledilo dogajanju in beležilo vsak klik na povezavo, vsak vnos podatkov, tako sam podatek, kot polje, kamor je bil podatek vnesen, vsak klik na gumb, itd. Snemanje zaključimo s pritiskom tipk `ctrl+c` v ukazni vrstici. Program bo zaključil snemanje in ustvaril dve datoteki.

Testiranje in poročila

Datoteka `prviTest.conf` vsebuje vse nastavitve, ki so povezane s testnim primerom. V njej so navedeni:

- URL strani, ki jo testiramo,

- prikaz poročila: poročilo je lahko prikazano v konzoli, shranjeno v datoteko ali oboje,
- nastavitve poti do datotek s poročilom in rezultatom,
- seznam HTTP odgovorov, ki se smatrajo za uspešne,
- čas premorov med vsako poslano zahtevo ter med vsakim testom ali ciklom.

Datoteka `test_prviTest.py` pa vsebuje posnet scenarij testa. Sedaj je bilo potrebno dodati samo še vrstice, kjer bomo preverjali dobljene rezultate s pričakovanimi. Končni testni primer zgleda približno tako:

Koda 2.4: Testni primer

```
import unittest
from funkload.FunkLoadTestCase import FunkLoadTestCase
from webunit.utility import Upload
from funkload.utils import Data
class Prvitest(FunkLoadTestCase):
    def setUp(self):
        self.logd("setUp")
        self.server_url = self.conf_get('main', 'url')
    def test_prviTest(self):
        server_url = self.server_url
        self.get(server_url + "/",description="Get /")
        self.assert_('Log in' in self.getBody(), 'Napacna stran')
        self.post(server_url + "/login", params=[
            ['username', 'uporabnik'],
            ['password', 'gesslo'],
            ['autologin', 'on'],
            ['redirect', ''],
            ['query', ''],
            ['login', 'Log in']],
            description="Post /login")
```

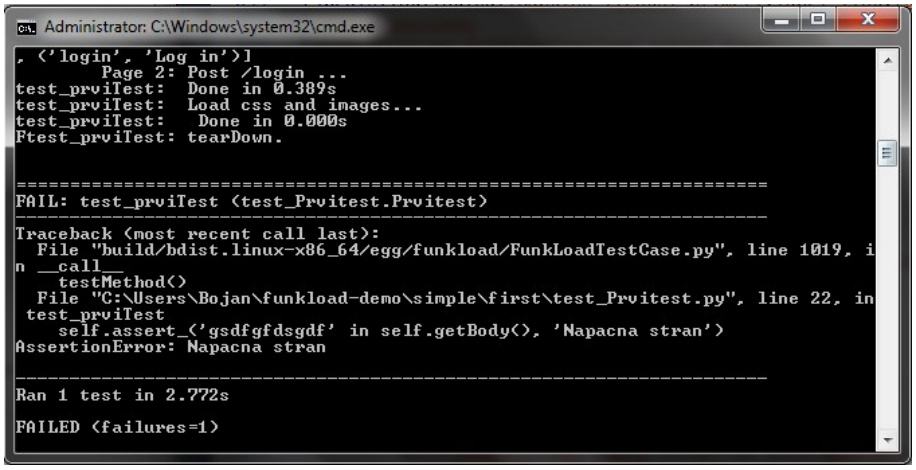
```
        self.assert_('Moj Tesni Forum' in self.getBody(), 'Napacna
            stran')
    def tearDown(self):
        self.logd("tearDown.\n")
if __name__ in ('main', '__main__'):
    unittest.main()
```

Iz kode 2.4 smo odstranili prazne vrstice in vrstice s komentarji. Iz programa je razvidno, da se FunkLoad izvaja znotraj Python-a. Poleg tega za delovanje uporablja Unittest [3] in Webunit [12]. Med delovanjem se v konzoli izpisuje trenutno stanje, čas izvajanja vsakega nalaganja nove strani in rezultati. Vse to se zapiše tudi v datoteko `prviTest-test.log` in zraven ustvari še `prviTest-test.xml` z dodatnimi informacijami.

Koda 2.5: Poročilo uspešnega testa

```
2015-07-19 23:11:04,434 DEBUG test_prviTest: Starting
    ------------------------------------
   XXX the test case description
2015-07-19 23:11:04,436 DEBUG test_prviTest: setUp
2015-07-19 23:11:04,437 DEBUG test_prviTest: GET:
    http://www.mojtestniforum.net/
   Page 1: Get / ...
2015-07-19 23:11:04,799 DEBUG test_prviTest: Done in 0.363s
2015-07-19 23:11:04,801 DEBUG test_prviTest: Load redirect link:
    http://www.mojtestniforum.net/login
2015-07-19 23:11:05,092 DEBUG test_prviTest: Done in 0.291s
2015-07-19 23:11:05,092 DEBUG test_prviTest: Load css and images...
2015-07-19 23:11:08,049 DEBUG test_prviTest: Done in 2.950s
2015-07-19 23:11:08,051 DEBUG test_prviTest: POST:
    http://www.mojtestniforum.net//login [('username',
    'uporabnik'), ('password', 'gesslo'), ('autologin', 'on'),
    ('redirect', ''), ('query', ''), ('login', 'Log in')]
   Page 2: Post /login ...
```

```
2015-07-19 23:11:08,427 DEBUG test_prviTest: Done in 0.376s
2015-07-19 23:11:08,427 DEBUG test_prviTest: Load css and images...
2015-07-19 23:11:08,433 DEBUG test_prviTest: Done in 0.000s
2015-07-19 23:11:08,434 DEBUG test_prviTest: tearDown.
```



Slika 2.5: Poročilo neuspešnega testa FunkLoad

Poročilo, ki je vidno na sliki 2.5, prikazuje izpis v primeru neuspešnega testa. Tega izpisa nam ni uspelo izpisati v datoteko, prav tako se poročilo napake ne izpiše v datoteko `prviTest-test.log`. Iz poročila so razvidni vrstica in ime datoteke testa ter izpis besedila, ki smo ga pri ukazu `self.assert_` vnesli.

Končna ocena

- Domača stran, dokumentacija in navodila - Ocena: 4

  Pozitivno: Ukazi so predstavljeni z opisom in praktično rabo, dokumentacija s spletne strani je dostopna tudi v PDF formatu, stran je urejena s preglednim kazalom, naveden je seznam najpogostejših vprašanj.

- Namestitev - Ocena: 5

Pozitivno: Namestitev je hitra in enostavna, postopek namestitve je naveden na strani skupaj z rešitvami na morebitne napake.

- Izdelava testnih primerov - Ocena: 5

  Pozitivno: Omogoča snemanje dogajanja na spletnem brskalniku, kar nato pretvori v test, ukazi za preverjanje temeljijo na Unittest-ih.

- Testiranje - Ocena: 4

  Pozitivno: Testiranje poteka hitro, v konzoli se izpisuje trenutno stanje ter koliko časa je potreboval za odprtje strani (skupaj s slikami in oblikovalnimi predlogami).

- Poročila - Ocena: 4

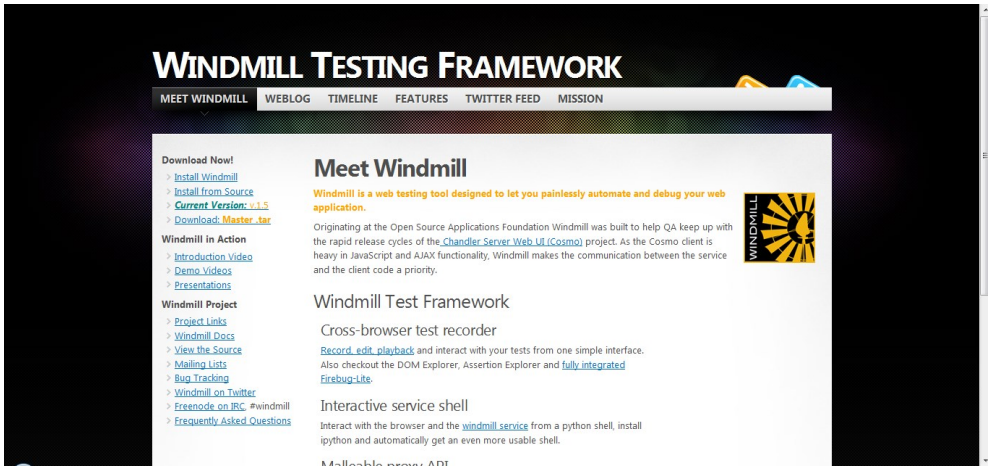
  Pozitivno: Poročilo izpiše tudi v `.log` datoteko.

  Negativno: Napake ne izpiše v `.log` datoteko.

Testno orodje je enostavno za uporabo. S snemanjem preko spletnega brskalnika omogoča hitro pripravo testov in odličen nadzor nad njim. Tudi posneta koda je enostavna za razumevanje in dopolnjevanje. Orodje je redno nadgrajevano z odpravljanjem napak in dodajanjem novih funkcij. Poročanje napak ter iskanje rešitev in nasvetov poteka preko spletne strani GitHub [15], kjer so razvijalci in uporabniki aktivni.

2.6 Orodje Windmill

Windmill je testno orodje, ki avtomatizira testiranje preko spletnega brskalnika.[16] Orodje posname vnose in klike, ki jih preko spletnega brskalnika (Firefox, Internet Explorer, Chrome, Safari) vnašamo oz. opravljamo. Tako posnet testni primer zaženemo znotraj brskalnika in opazujemo, kako orodje izvaja testni primer kot bi to opravljal uporabnik. Orodje namreč testiranje aplikacije opravlja preko spletnega brskalnika.



Slika 2.6: Spletna stran orodja Windmill

2.6.1 Pregled in ocenjevanje

Domača stran, dokumentacija in navodila

Spletna stran orodja (Slika 2.6) je pregledno izdelana. Na njej najdemo video predstavitve orodja, predstavitvene prosojnice in navodila za namestitev. Navedene so tudi povezave do seznama najpogostejših vprašanj, seznama elektronskih sporočil, strani za sledenje napakam in socialnih omrežij.

Namestitev

Postopek namestitve je naveden na spletni strani orodja, ki zahteva še nekaj dodatnih programov, ki jih moramo namestiti pred namestitvijo Windmill-a.

Problem

Prvi problem z orodjem je nastal ob prvem zagonu, saj Windmill ni našel brskalnikov Firefox in Chrome, pri Internet Explorerju (kratica IE) pa je javil napako. Za napako pri IE smo našli rešitev na spletu, natančneje na njihovi strani za sledenje napakam (ang. bug tracking), ki pa je bila preveč negativna. Potrebno je bilo namreč odstraniti Python 2.7 verzijo in namesti

2.6 verzijo.

Pri Firefox-u je nastal problem, ker ga na svojem računalniku nimamo nameščenega v mapi Programske datoteke (ang. Program files), ampak je nameščen na drugi particiji diska. Zato smo namestili Chrome na privzeto mesto, vendar ga Windmill še vedno ni uspel najti. S pregledovanjem kode smo uspeli najti, kje so navedene poti do posameznih brskalnikov. Tu nam je uspelo nastaviti pravilno pot do Chrome brskalnika in Windmill ga je uspešno zaznal. Enako smo poskušali nastaviti še za Firefox, vendar je bilo v kodi programa veliko nerazumljive kode, ki se je navezovala s Firefoxom, obdane s komentarji, ki so preprečevali izvajanje.

Windmill smo tako lahko uspešno s Chrome brskalnikom zagnali iz ukazne vrstice. Tu je nastal drugi problem, Windmill se ni zagnal do konca. Med nalaganjem se ustavi, vendar ni javil nobene napake. Na spletu nismo našli rešitve, saj so razvijalci neaktivni. Večina poročil o napakah je brez odgovorov. Prav tako je bilo brez odgovora poročilo o napaki, ki je bila zelo podobna naši.

Končna ocena

- Domača stran, dokumentacija in navodila - Ocena: 5

  Pozitivno: Pregledana stran, enostavna za navigacijo, video predstavitev.

- Namestitev - Ocena: 2

  Pozitivno: Namestitev je hitra in enostavna, postopek namestitve je naveden na strani skupaj s številnimi dodatnim orodij.

  Negativno: Ob namestitvi orodje ni primerno za Windows okolje.

2.7 Končni rezultati

Twill je med vsemi orodji najbolj enostaven za uporabo, saj vsebuje kratka in enostavna imena metod. S svojo enostavnostjo je idealen za vse, ki se

	Twill	**Webunit**	**FunkLoad**	**Windmill**
Domača stran	3	2	4	5
Namestitev	5	2	5	2
Izdelava testnih primerov	3	/	5	/
Testiranje	4	/	4	/
Poročila	3	/	4	/
Povprečna ocena	**3,6**	**2**	**4,4**	**3,5**

Tabela 2.1: Končne ocene

prvič spuščajo na področje testiranja spletnih aplikacij ter za uporabo pri manjših projektih. Pri večjih projektih pa postane pisanje testnih primerov zelo zamudno.

Najbolje ocenjeno (tabela 2.1) testno orodje je FunkLoad, saj je enostavno in pregledno za uporabo. Testnih primerov ni potrebno pisati na roko, saj jih orodje sestavlja s pomočjo snemanja našega izvajanja v brskalniku. Tako posnet testni primer le še dopolnimo. To nam zelo olajša delo na primer, ko testiramo izpolnjevanje večjega števila obrazcev (registracija v aplikacijo), saj nam ni potrebno iskati imen obrazcev. Nadzor in izvajanje testov ter poročila so enostavno berljiva in ker izpisuje poročilo testa tudi v datoteko, ni dolžina omejena s količino znakov v konzoli. Pomanjkljiv je le izpis pri napakah, saj le teh ne izpisuje v datoteke.

Poglavje 3

Orodja za pregledovanje kode

V tem poglavju se bomo omejili na programe za pregledovanje programske kode jezika Python. Python za razliko od večine programskih jezikov nima prevajalnika. Kodo napišemo in shranimo v `.py` datoteko. To datoteko nato izvedemo z ukazom `python` v konzoli in program se že izvaja. Pri drugih jezikih prevajalniki služijo tudi kot orodje za pregledovanje napak v kodi, pri Pythonu pa potrebujemo dodatna orodja, ki kodo pregledajo in javijo napake.

3.1 Postopek testiranja in ocenjevanja

Namestitev

Tu si bomo ogledali sam postopek namestitve ter priprave orodja pregledovanje kode. Ocenjevali bomo po naslednjih glavnih kriterijih:

- Postopek namestitve mora biti opisan na spletni strani. Če za uporabo orodja potrebujemo namestiti še dodatne pakete, morajo biti navedeni skupaj z navodili za namestitev.
- Orodje mora po namestitvi biti pripravljeno za delovanje. V nasprotnem primeru mora orodje nuditi enostavno prilagoditev. Postopek in razlogi za prilagoditev morajo biti navedeni na spletni strani.

Uspešnost pregledovanja

Testna orodja bodo pregledovala dve različni kodi in poskušala opozoriti na določene napake, ki smo jih namerno ustvaril v kodi. Koda programa (3.1) vsebuje nekatere najpogostejše napake, iz kode programa (3.2) pa bomo izmenično odstranjevali dvopičja, oklepaje, vejice in podobno.

Ocenjevali bomo po naslednjih glavnih kriterijih:

- Orodje mora odkriti vse napake in nepravilnosti.
- Lokacije napak morajo biti natančno navedene.

Koda 3.1: Program 1

```
import sys
import unittest

class Dog:
    def __init__(self, pasma, spol, starost, ime):
        self.pasma = pasma
        self. spol = spol
        self.starost = starost
        self .ime = ime
        self.teza=self.getTeza(ime, pasma)
        self.lastnik = oseba.getIme(ime);
    def getTeza(this):
        return 15
    def izpis():
        print "zelo dolga vrsticaaaa ......................
            ................. .....................
            ................. ...."
```

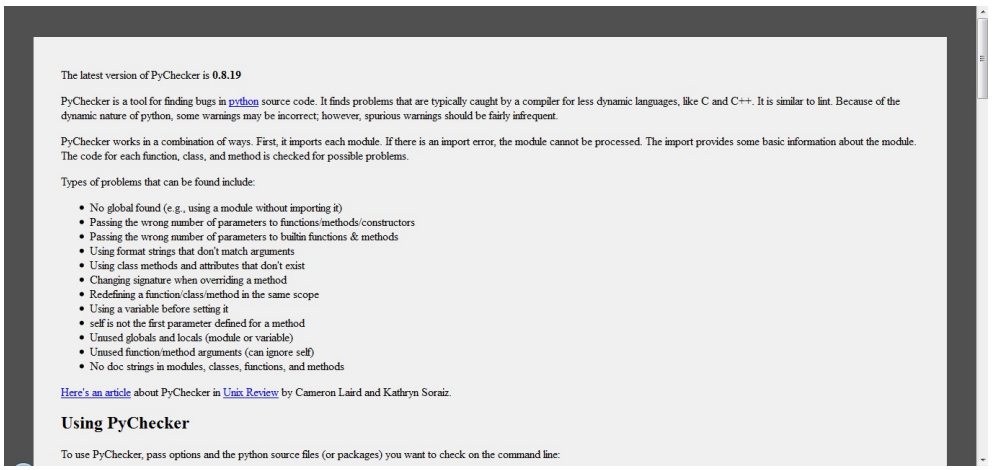
Koda 3.2: Program 2

```
def funkcija(a, b):
    return a+b
```

```
print funkcija(5, 7)
print 'hello'
```

3.2 Orodje PyChecker

Ob zagonu orodje PyChecker najprej uvozi vsak modul in nato preveri vsak razred, funkcijo in metodo za napake.[17] V primeru, ko ne uspe uvoziti kode, javi napako, iz katere je razvidno, zaradi česa je prišlo do napake. Zadnja verzija je 0.8.19 in je bila izdana 8. januarja 2011 (Slika 3.1).


The latest version of PyChecker is **0.8.19**

PyChecker is a tool for finding bugs in python source code. It finds problems that are typically caught by a compiler for less dynamic languages, like C and C++. It is similar to lint. Because of the dynamic nature of python, some warnings may be incorrect; however, spurious warnings should be fairly infrequent.

PyChecker works in a combination of ways. First, it imports each module. If there is an import error, the module cannot be processed. The import provides some basic information about the module. The code for each function, class, and method is checked for possible problems.

Types of problems that can be found include:

- No global found (e.g., using a module without importing it)
- Passing the wrong number of parameters to functions/methods/constructors
- Passing the wrong number of parameters to builtin functions & methods
- Using format strings that don't match arguments
- Using class methods and attributes that don't exist
- Changing signature when overriding a method
- Redefining a function/class/method in the same scope
- Using a variable before setting it
- self is not the first parameter defined for a method
- Unused globals and locals (module or variable)
- Unused function/method arguments (can ignore self)
- No doc strings in modules, classes, functions, and methods

Here's an article about PyChecker in Unix Review by Cameron Laird and Kathryn Soraiz.

Using PyChecker

To use PyChecker, pass options and the python source files (or packages) you want to check on the command line:

Slika 3.1: Spletna stran orodja PyChecker

Namestitev

PyChecker lahko namestimo preko ukaza `easy_ install`, vendar pa privzeta namestitev ni primerna za Windows sistem. Ob namestitvi se v mapo `Scripts` namesti datoteka `pychecker.bat`, katere vsebino lahko vidimo v Kodi 3.3. Služi za izvajanje orodja v Windowsu, vendar ne deluje. Poskušali smo spreminjati vsebino datoteke glede na napake, ki nam jih je izpisoval operacijski sistem v konzoli, vendar ni bilo uspeha. Na koncu smo celotno vsebino spremenili z dejanskimi potmi do pravilnih datotek (razvidno v

kodi 3.4) in orodje je končno pričelo delovati brez napak.

Koda 3.3: Original pychecker.bat

```
#!D:\Python27\python.exe
# EASY-INSTALL-SCRIPT: 'pychecker==0.8.19','pychecker.bat'
__requires__ = 'pychecker==0.8.19'
__import__('pkg_resources').run_script('pychecker==0.8.19',
    'pychecker.bat')
```

Koda 3.4: Popravljena pychecker.bat

```
D:\Python27\python.exe
    D:\Python27\Lib\site-packages\pychecker-0.8.19-py2.7.egg\
    pychecker\checker.py %*
```

Uspešnost pregledovanja

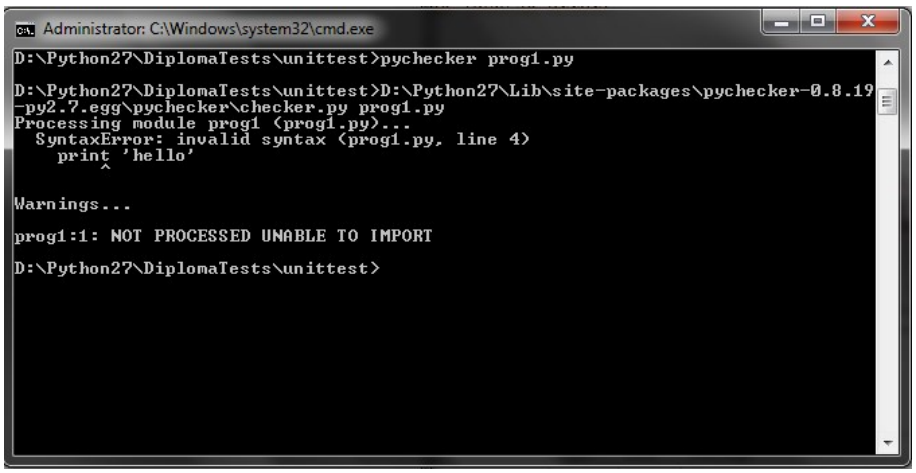
Poročilo (3.5) se izpisuje v konzoli. Vsaka vrstica je sestavljena iz imena programa, številke vrstice napake in opisa napake. Orodje je odkrilo 6 napak, ki so lepo opisane in razumljive. Navedlo je neuporabljene uvožene knjižnice (vrstici 1 in 2), napačno število argumentov pri klicu metode (vrstica 10), manjkajoči globalni razred (vrstica 11) ter dvakrat napaka `self` argumenta (vrstici 12 in 14).

Koda 3.5: Poročilo

```
Warnings...

prog.py:1: Imported module (sys) not used
prog.py:2: Imported module (unittest) not used
prog.py:10: Invalid arguments to (getTeza), got 2, expected 0
prog.py:11: No global (oseba) found
prog.py:12: self is not first method argument
prog.py:14: No method arguments, should have self as argument
```

Pri manjkajočih oklepajih, vejicah in podobno, orodje javi napako zelo podobno, kot bi jo javil Python, če bi program poskušali izvesti (Slika 3.2). PyChecker se je ob prvi taki usodni napaki ustavil in ni nadaljeval testiranja, kar je tudi navedel med opozorili.



Slika 3.2: Poročilo neuspešnega testa PyChecker

Končna ocena

- Ocena namestitve: 3
- Ocena delovanja: 3

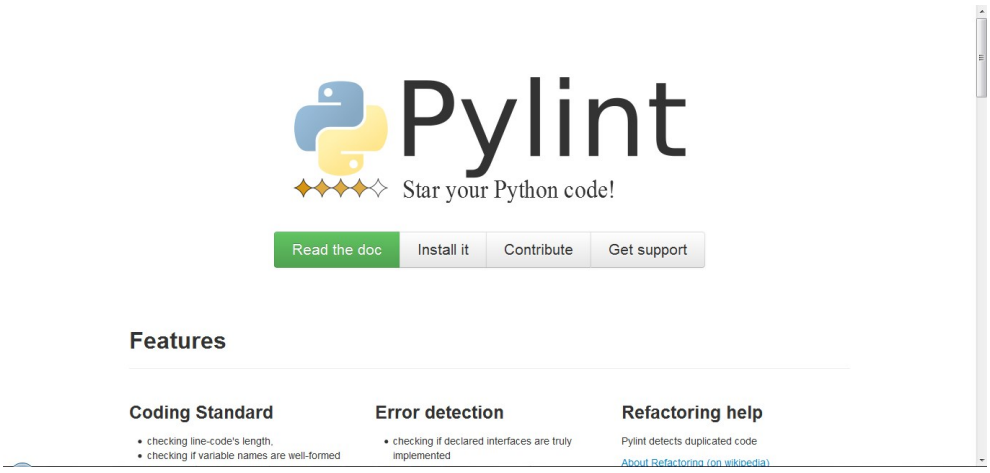
Orodje odkriva vse ključne napake, zaradi katerih končni program ne bo deloval pravilno ter manjše napake, ki bi le upočasnile izvajanje programa (uvažanje neuporabljenih knjižnic). Iz poročil sta jasno razvidni lokacija in opis napake, tako da omogoča hitro odpravo napak.

Pri usodnih napakah bo odkril le prvo napako, jo javil in zaključil delovanje. Pri določanju lokacije napake je orodje prav tako nezanesljivo kot Python. V primeru, da smo odstranili oklepaj na koncu klica funkcije (`print funkcija(5, 7`), je kot lokacijo napake navedel konec ukaza `print` v naslednji vrstici.

Velika pomanjkljivost orodja PyChecker je dejstvo, da mora program uvoziti in izvesti. V primeru, ko program vsebuje dolge `for` zanke, bo testiranje potekalo zelo počasi, saj se bo moral program najprej izvesti in šele na to ga bo orodje lahko testiralo.

3.3 Orodje Pylint

Pylint je orodje, ki je zasnovano na orodju PyChecker (poglavje 3.2), ki pa poleg iskanja napak, pregleduje kodo še za tako imenovanim lepopisjem.[18] Pregleduje tudi nepotrebne presledke, manjkajoče presledke, dolžino vrstic, nepotrebne znake, komentarje, ki so namenjeni dokumentaciji, in podobno. Zaradi tega je odlično nadomestilo orodja PyChecker.[19] Slika (3.3) prikazuje spltno stran orodja.



Slika 3.3: Spletna stran orodja Pylint

Namestitev

Pylint lahko namestimo preko ukaza `pip install`, vendar pa, podobno kot pri orodju PyChecker, privzeta namestitev v mapo `Scripts` ni primerna za Windows sistem. Ob zagonu kozola sporoči, da datoteka ne obstaja. Ponovno smo morali preuredi datoteko `pylint.bat` v mapi `Scripts`. Ta je namreč

poskušala v isti mapi zagnati datoteko `pylint` in ne datoteke `pylint.exe`, ki se nahaja v mapi in izvaja program. Potrebno je bilo zadnji vrstici za navednice dodati `.exe` (Koda 3.6), pa je orodje pričelo delovati.

Koda 3.6: Končni izgled pylint.bat

```
@echo off
rem Use python to execute the python script having the same name
    as this batch
rem file, but without any extension, located in the same directory
    as this
rem batch file
python "%~dpn0".exe %*
```

Uspešnost pregledovanja

Pylint je našel veliko več napak kot PyChecker. Našel je vseh 6 osnovnih napak, nekatere izmed njih je dopolnil ter dodal še svoje lepopisne napake. Kot lepopisne napake je navedel nepotrebne presledke v 3. vrstici (iz testne kode 3.1 sicer ni razvidno, vendar so v 3. vrstici trije presledki), manjkajoče presledke za vejicami in pred znakom za enakost, predolgo vrstico ter manjkajoče komentarje za dokumentacijo (ang. docstring). Našel pa ni nepotrebnih presledkov pri pikah, ki pri Pythonu ne povzročajo težav, vendar bi tudi to moralo spadati pod lepopisje.

Poročilo 3.7 je le del poročila, ki ga Pylint izpiše v konzolo. Pri navajanju napak je natančnejši, saj poleg vrstice javi še stolpec, v katerem se napaka prične, vendar pri tem ignorira začetne zamike. Na začetku navede še kategorijo napake: E označuje resne napake, W označuje manjše napake, zaradi katerih bo program vseeno deloval pravilno, C pa označuje napake povezane z dogovorjenim pisanjem kode in lepopisjem, R pa napake, ki kršijo tako imenovano dobro prakso (v poročilu navaja, da bi lahko metoda bila samo funkcija, saj se ne sklicuje na dani razred).

Koda 3.7: Poročilo

```
C:  3, 0: Trailing whitespace (trailing-whitespace)
C:  5, 0: Exactly one space required after comma
    def __init__(self, pasma,spol,starost, ime):
                            ^ (bad-whitespace)
C:  5, 0: Exactly one space required after comma
    def __init__(self, pasma,spol,starost, ime):
                               ^ (bad-whitespace)
C: 10, 0: Exactly one space required around assignment
        self.teza=self.getTeza(ime, pasma)
                ^ (bad-whitespace)
W: 11, 0: Unnecessary semicolon (unnecessary-semicolon)
C: 15, 0: Line too long (126/100) (line-too-long)
C:  1, 0: Missing module docstring (missing-docstring)
C:  4, 0: Missing class docstring (missing-docstring)
C:  4, 0: Old-style class defined. (old-style-class)
E: 10,18: Too many positional arguments for method call
    (too-many-function-args)
E: 11,23: Undefined variable 'oseba' (undefined-variable)
C: 12, 4: Invalid method name "getTeza" (invalid-name)
C: 12, 4: Missing method docstring (missing-docstring)
E: 12, 4: Method should have "self" as first argument
    (no-self-argument)
R: 12, 4: Method could be a function (no-self-use)
C: 14, 4: Missing method docstring (missing-docstring)
E: 14, 4: Method has no argument (no-method-argument)
R: 14, 4: Method could be a function (no-self-use)
W:  1, 0: Unused import sys (unused-import)
W:  2, 0: Unused import unittest (unused-import)
```

Pri manjkajočih oklepajih, vejicah in podobno, orodje javi napako sintakse, vendar za razliko od Pythona, ne izpiše celotne vrstice in označbe napake, temveč navede le vrstico napake (Slika 3.4).

Slika 3.4: Poročilo neuspešnega testa Pylint

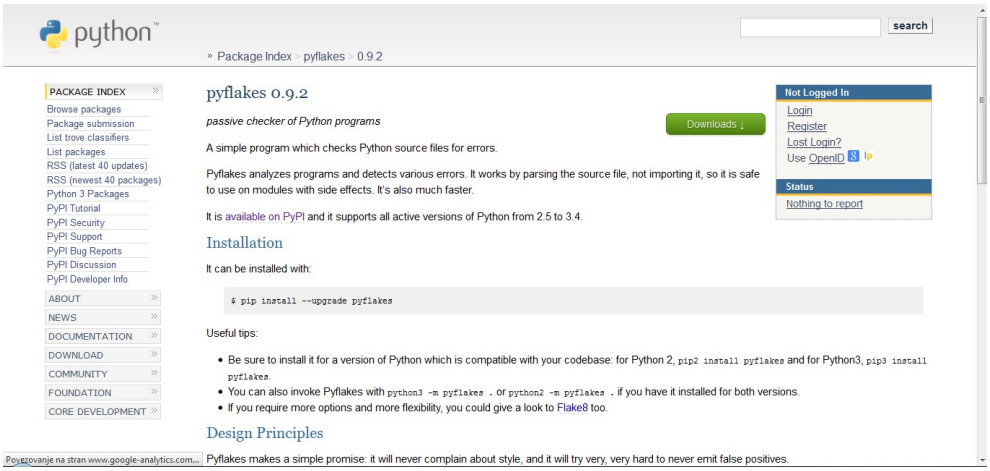
Končna ocena

- Ocena namestitve: 4
- Ocena delovanja: 4

Pylint je odlična nadgradnja orodja PyChecker in je zelo primerna za vse, ki bi želeli, da bi bila vaša koda pregledno in lepo zapisana. Iz poročil sta jasno razvidni lokacija in opis napake, tako da omogoča hitro odpravo napak. Prav tako je v drugem delu poročila navedena analiza kode, ki izpisuje število razredov, funkcij, metod, komentarjev ter število napak glede na kategorijo. Na koncu je še podana ocena kode.

Pylint se ob prvi usodni napaki ustavi in ne nadaljuje testiranja. Pri določanju lokacije napake je orodje prav tako nezanesljivo kot Python, saj v primeru, da smo odstranili oklepaj na koncu klica funkcije (`print funkcija(5, 7`) je kot lokacijo napake navedel 4. vrstico (Slika 3.4).

Pylint testiranih programov, ne izvaja, tako da se čas testiranja s številom ponovitev zanke ne bo podaljševal.



Slika 3.5: Spletna stran orodja Pyflakes

3.4 Orodje Pyflakes

Orodje Pyflakes je tudi zasnovano na orodju PyCheck (poglavje 3.2) in enako preverja samo osnovne napake.[20] Glavna razlika je, da Pyflakes za svoje preverjanje ne izvaja programa, ampak ga pregleduje kot besedilo. To mu omogoča hitro in varno delovanje, saj se programi ne izvajajo.

Namestitev

Pyflakes lahko namestimo preko ukaza `pip install`, ki je opisan na spletni strani orodja (Slika 3.5). Ob namestitvi je orodje že pripravljeno za delovanje, brez dodatnih posegov v kodo programa.

Uspešnost pregledovanja

Pyflakes je našel samo 3 napake. Našel je napaki o nepotrebnih uvoženih knjižnicah in manjkajočem razredu, ni pa odkril napake o napačnem številu argumentov. Poročilo 3.8 je zelo podobno poročilu PyChecker-ja, spremenjen je le opis napake.

Koda 3.8: Poročilo

```
prog.py:1: 'sys' imported but unused
prog.py:2: 'unittest' imported but unused
prog.py:11: undefined name 'oseba'
```

Pri manjkajočih oklepajih, vejicah in podobno, orodje javi napako zelo podobno kot Python, če bi program poskušali zagnati (Slika 3.6). Pyflakes se ob prvi taki usodni napaki ustavi in ne nadaljuje testiranja.

Slika 3.6: Poročilo neuspešnega testa Pyflakes

Končna ocena

- Ocena namestitve: 5
- Ocena delovanja: 2

Glavna prednost orodja Pyflakes je njegova hitrost in možnost testiranja brez zagona kode. Vendar ga dejstvo, da ne najde vseh napak, naredi manj zanesljivega za uporabo.

Pri usodnih napakah, bo enako kot Python odkril le prvo napako, jo javil in zaključil delovanje. Pri določanju lokacije napake je orodje prav tako nezanesljivo kot Python. V primeru, da smo odstranili oklepaj na koncu klica funkcije (`print funkcija(5, 7`), je kot lokacijo napake navedel konec ukaza `print` v naslednji vrstici.

3.5 Končni rezultati

	PyChecker	**Pylint**	**Pyflakes**
Namestitev	3	4	5
Delovanje	3	4	2
Povprečna ocena	**3**	**4**	**3,5**

Tabela 3.1: Končne ocene

Kot orodje za pregledovanje kode je bilo najbolje ocenjeno (tabela 3.1) Pylint. Zanj smo se odločili, ker ne deluje samo kot testno orodje za preverjanje standardnih napak, ampak ga lahko uporabljamo kot pomoč pri učenju pisanja lepo oblikovane kode. Prav to je velika prednost pri projektih, ki vključujejo večje število ljudi, saj lahko vsak razume tudi kodo, ki je ni sam napisal.

Orodji PyChecker in Pyflakes ne priporočamo, saj imata obe nekaj negativnih lastnosti. Pri orodju PyChecker je največja negativnost ta, da program najprej poskuša izvesti. Zaradi tega je problem testirati programe z dolgimi zankami, programe, ki ustvarjajo določene datoteke, baze itd., saj se morajo programi uspešno izvesti in zaključiti. Pri orodju Pyflakes je ta napaka sicer odpravljena, vendar pa orodje ne najde vseh napak (iz primera v poglavju 3.4 je razvidno, da napačnega števila argumentov ni našel).

Poglavje 4

Zaključek

Cilj diplomskega dela je bil sestaviti nabor testnih orodij iz različnih področij, ki so najprimernejša za uporabo.

Med orodij za testiranje spletnih aplikacij smo izbrali orodje FunkLoad, ki je enostaven za uporabo. Namestitev je preprosta in brez zapletov. Povratne informacije med izvajanjem testov in končna poročila so jasna in natančna.

Orodje Pylint smo izbrali kot najprimernejše orodje na področju pregledovanja napak programske kode. Deluje hitro, ne izvaja kode in spodbuja uporabnika k lepemu in preglednemu pisanju programske kode.

Končni nabor orodij:

- orodje FunkLoad
- orodje Webunit (kot dodatna knjižnica FunkLoad-a)
- orodje Unittest (kot dodatna knjižnica FunkLoad-a)
- orodje Pylint

Pregledali smo le dve področji testiranja in le nekaj orodij iz obeh področij. Potrebno bi bilo še pregledati druga področja testiranja, na primer obremenitveno in stresno testiranje (ang. load and stress testing [21]) in orodja za pregledovanje pokritosti kode (ang. code coverage tools [22]. Tudi

na pregledanih dveh področjih so ostala nekatera nepregledana orodja. Potrebno bi bilo pregledati še orodja, ki so izdelana za testiranje določenih spletnih ogrodij (ang. web frameworks).

Literatura

[1] G. D. Everett, R. McLeod Jr.. "Software Testing: Testing Across the Entire Software Development Life Cycle", *John Wiley & Sons, Inc.*, 2007, str. 159.

[2] M. Anders. "Python 3 Web Development: Beginner's Guide", *Packt Publishing Ltd*, 2011.

[3] Testna knjižnica Unittest. [Online]. Dosegljivo: https://docs.python.org/3/library/unittest.html. [Dostopano 5. 8. 2015].

[4] Likertova lestvica. [Online]. Dosegljivo: https://en.wikipedia.org/wiki/Likert_scale. [Dostopano 27. 8. 2015].

[5] Sistematika Python testnih orodij. [Online]. Dosegljivo: https://wiki.python.org/moin/PythonTestingToolsTaxonomy. [Dostopano 5. 8. 2015].

[6] Testno orodje JUnit. [Online]. Dosegljivo: http://junit.org/. [Dostopano 5. 8. 2015].

[7] Testno orodje CUnit. [Online]. Dosegljivo: http://cunit.sourceforge.net/. [Dostopano 5. 8. 2015].

[8] Testno orodje NUnit. [Online]. Dosegljivo: http://www.nunit.org/. [Dostopano 5. 8. 2015].

[9] Testni programski jezik Twill. [Online]. Dosegljivo: http://twill.idyll.org/. [Dostopano 5. 8. 2015].

[10] Programski jezik JavaScript. [Online]. Dosegljivo: https://www.javascript.com/. [Dostopano 5. 8. 2015].

[11] C. Titus Brown, G. Gheorghiu, J. R. Huggins. "An Introduction to Testing Web Applications with twill and Selenium", *O'Reilly Media, Inc.*, 2007.

[12] Testno orodje Webunit. [Online]. Dosegljivo: http://mechanicalcat.net/tech/webunit/. [Dostopano 5. 8. 2015].

[13] Testno orodje FunkLoad. [Online]. Dosegljivo: http://funkload.nuxeo.org/. [Dostopano 5. 8. 2015].

[14] S. Hellegouarch. "CherryPy Essentials", *Packt Publishing Ltd.*, 2007, str. 213-218.

[15] GitHub. [Online]. Dosegljivo: https://github.com/. [Dostopano 5. 8. 2015].

[16] Testno orodje Windmill. [Online]. Dosegljivo: http://www.getwindmill.com/. [Dostopano 5. 8. 2015].

[17] Testno orodje PyChecker. [Online]. Dosegljivo: http://pychecker.sourceforge.net/. [Dostopano 5. 8. 2015].

[18] Testno orodje Pylint. [Online]. Dosegljivo: http://www.pylint.org/. [Dostopano 5. 8. 2015].

[19] M. L. Hetland. "Beginning Python: From Novice to Professional", *Apress*, 2008, str. 359-361.

[20] Testno orodje Pyflakes. [Online]. Dosegljivo: https://github.com/pyflakes/pyflakes. [Dostopano 5. 8. 2015].

[21] E. Vervaet, "The Definitive Guide to Spring Web Flow", *Apress*, 2008, str. 319-322.

[22] Orodja za pregled pokritosti kode. [Online]. Dosegljivo: http://c2.com/cgi/wiki?CodeCoverageTools. [Dostopano 27. 8. 2015].